What is claimed is:

1. A method for identifying errors in program code, comprising:

performing by a processor the steps including,

counting in the program code respective numbers of observances of at least one correctness rule by different code instances that relate to the at least one correctness rule, wherein each code instance has an associated counted number of observances of the correctness rule by the code instance;

counting in the program code respective numbers of violations of the at least one correctness rule by different code instances that relate to the at least one correctness rule, wherein each code instance has an associated counted number of violations of the correctness rule by the code instance;

determining for each code instance a respective likelihood of validity of the code instance as a function of the counted number of observances and counted number of violations, wherein the likelihood of validity indicates a relative likelihood that a related code instance is required to observe the correctness rule; and

outputting the violations in order of the likelihood of validity of a violated correctness rule.

2. The method of claim 1, wherein the determining step further comprises determining a likelihood of the validity of each code instance as a function of an expected ratio of observances to violations, the counted number of observances, and the counted number of violations.

3. The method of claim 2, wherein the determining step further comprises statistically ranking each violation according to a number of standard deviations away, a ratio of the counted number of observances to the counted number of violations is from the expected ratio.

4. The method of claim 3, wherein statistically ranking each violation includes determining a z statistic for proportions.

5. The method of claim 1, further comprising:

wherein a first correctness rule specifies that a variable must be protected by a lock before accessing the variable;

the step of counting an observance of the first correctness rule by a particular code instance includes identifying program code that locks a particular first variable followed by program code that accesses a particular second variable; and

the step of counting a violation of the first correctness rule by the particular code instance includes identifying program code that accesses the particular second variable where no preceding program code locks the particular first variable.

6. The method of claim 1, further comprising: wherein a first correctness rule specifies that invocation of a first function must not follow an invocation of a second function in the program code;

the step of counting an observance of the first correctness rule by a particular code instance includes identifying program code that includes a sequence of instructions that includes invocation of a particular second instruction and no previous invocation of a particular first function; and

the step of counting a violation of the first correctness rule by the particular code instance includes identifying program code that includes a sequence of instructions in which an invocation of a particular first function is present following invocation of a particular second instruction.

7. The method of claim 1, further comprising:

wherein a first correctness rule specifies that invocation of a first function must follow an invocation of a second function in the program code;

the step of counting an observance of the first correctness rule by a particular code instance includes identifying program code that includes a sequence of instructions that includes an invocation of a particular first function following invocation of a particular second instruction; and

the step of counting a violation of the first correctness rule by the particular code instance includes identifying program code that includes a sequence of instructions that includes an invocation of the particular second instruction without a previous invocation of the particular first function.

8. The method of claim 1, further comprising:

wherein a first correctness rule specifies that data returned from a first function must be tested for a status indication;

the step of counting an observance of the first correctness rule by a particular code instance includes identifying program code that includes a sequence of instructions that includes an invocation of a particular first function and a subsequent test of data returned from the particular first function; and

the step of counting a violation of the first correctness rule by the particular code instance includes identifying program code that includes a sequence of instructions that includes an invocation of a particular first function without a subsequent test of data returned from the particular first function.

9. An apparatus for identifying errors in program code, comprising:

means for counting in the program code respective numbers of observances of at least one correctness rule by different code instances that relate to the at least one correctness rule, wherein each code instance has an associated counted number of observances of the correctness rule by the code instance;

means for counting in the program code respective numbers of violations of the at least one correctness rule by different code instances that relate to the at least one correctness rule, wherein each code instance has an associated counted number of violations of the correctness rule by the code instance;

means for determining for each code instance a respective likelihood of the validity as a function of the counted number of observances and counted number of violations, wherein the likelihood of validity indicates a relative likelihood that a related code instance is required to observe the correctness rule; and

means for outputting the violations in order of the likelihood of validity of a violated correctness rule.

10. A system for identifying errors in program code, comprising:

a data processing arrangement;

an analyzer hosted on the data processing arrangement, the analyzer configured to,

count in the program code respective numbers of observances of at least one correctness rule by different code instances that relate to the at least one correctness rule, wherein each code instance has an associated counted number of observances of the correctness rule by the code instance;

count in the program code respective numbers of violations of the at least one correctness rule by different code instances that relate to the at least one correctness rule, wherein each code instance has an associated counted number of violations of the correctness rule by the code instance;

determine for each code instance a respective likelihood of validity of the code instance as a function of the counted number of observances and counted number of violations, wherein the likelihood of validity indicates a relative likelihood that a related code instance is required to observe the correctness rule; and

output the violations in order of the likelihood of validity of a violated correctness rule.

11. The system of claim 10, wherein the analyzer is further configured to, in determining of likelihood of validity, determine a likelihood of the validity of each code instance as a function of an expected ratio of observances to violations, the counted number of observances, and the counted number of violations.

12. The system of claim 11, wherein the analyzer is further configured to, in determining of likelihood of validity, statistically rank each violation according to a number of standard deviations away, a ratio of the counted number of observances to the counted number of violations is from the expected ratio.

13. The system of claim 12, wherein the analyzer is further configured to, in statistically ranking each violation, determine a *z* statistic for proportions.

14. The system of claim 10, further comprising:

wherein the analyzer is configured to count code instances of a first correctness rule that specifies that a variable must be protected by a lock before accessing the variable;

in counting an observance of the first correctness rule by a particular code instance, the analyzer is configured to identify program code that locks a particular first variable followed by program code that accesses a particular second variable; and

in counting a violation of the first correctness rule by the particular code instance, the analyzer is configured to identify program code that accesses the particular second variable where no preceding program code locks the particular first variable.

15. The system of claim 10, further comprising:

wherein the analyzer is configured to count code instances of a first correctness rule specifies that invocation of a first function must not follow an invocation of a second function in the program code;

in counting an observance of the first correctness rule by a particular code instance, the analyzer is configured to identify program code that includes a sequence of instructions that includes invocation of a particular second instruction and no previous invocation of a particular first function; and

in counting a violation of the first correctness rule by the particular code instance, the analyzer is configured to identify program code that includes a sequence of instructions in which an invocation of a particular first function is present following invocation of a particular second instruction.

16. The system of claim 10, further comprising:

wherein the analyzer is configured to count code instances of a first correctness rule specifies that invocation of a first function must follow an invocation of a second function in the program code;

in counting an observance of the first correctness rule by a particular code instance, the analyzer is configured to identify program code that includes a sequence of instructions that includes an invocation of a particular first function following invocation of a particular second instruction; and

in counting a violation of the first correctness rule by the particular code instance, the analyzer is configured to identify program code that includes a sequence of instructions that includes an invocation of the particular second instruction without a previous invocation of the particular first function.

17. The system of claim 10, further comprising:

wherein the analyzer is configured to count code instances of a first correctness rule specifies that data returned from a first function must be tested for a status indication;

in counting an observance of the first correctness rule by a particular code instance, the analyzer is configured to identify program code that includes a sequence of instructions that includes an invocation of a particular first function and a subsequent test of data returned from the particular first function; and

in counting a violation of the first correctness rule by the particular code instance, the analyzer is configured to identify program code that includes a sequence of instructions that includes an invocation of a particular first function without a subsequent test of data returned from the particular first function.

18. An article of manufacture, comprising:

an electronically readable medium configured with instructions for causing a processor to perform the steps including,

counting in the program code respective numbers of observances of at least one correctness rule by different code instances that relate to the at least one correctness rule, wherein each code instance has an associated counted number of observances of the correctness rule by the code instance;

counting in the program code respective numbers of violations of the at least one correctness rule by different code instances that relate to the at least one correctness rule, wherein each code instance has an associated counted number of violations of the correctness rule by the code instance;

determining for each code instance a respective likelihood of the validity as a function of the counted number of observances and counted number of violations, wherein the likelihood of validity indicates a relative likelihood that a related code instance is required to observe the correctness rule; and

outputting the violations in order of the likelihood of validity of a violated correctness rule.

19. The article of manufacture of claim 18, wherein the electronically readable medium is further configured with instructions for causing a processor, in determining a likelihood of validity, to perform the step comprising determining a likelihood of the validity of each code instance as a function of an expected ratio of observances to violations, the counted number of observances, and the counted number of violations.

20. The article of manufacture of claim 19, wherein the electronically readable medium is further configured with instructions for causing a processor, in determining a likelihood of validity, to perform the step comprising statistically ranking each violation according to a number of standard deviations away, a ratio of the counted number of observances to the counted number of violations is from the expected ratio.

21. The article of manufacture of claim 20, wherein the electronically readable medium is further configured with instructions for causing a processor, in statistically ranking each violation, to perform the step comprising determining a z statistic for proportions.

22. The article of manufacture of claim 18, wherein a first correctness rule specifies that a variable must be protected by a lock before accessing the variable, and the electronically readable medium is further configured with instructions for causing a processor to perform the steps comprising:

in counting an observance of the first correctness rule by a particular code instance, identifying program code that locks a particular first variable followed by program code that accesses a particular second variable; and

in counting a violation of the first correctness rule by the particular code instance, identifying program code that accesses the particular second variable where no preceding program code locks the particular first variable.

23. The article of manufacture of claim 18, wherein a first correctness rule specifies that invocation of a first function must not follow an invocation of a second function in the program code, and the electronically readable medium is further configured with instructions for causing a processor to perform the steps comprising:

in counting an observance of the first correctness rule by a particular code instance, identifying program code that includes a sequence of instructions that includes invocation of a particular second instruction and no previous invocation of a particular first function; and

in counting a violation of the first correctness rule by the particular code instance, identifying program code that includes a sequence of instructions in which an invocation of a particular first function is present following invocation of a particular second instruction.

24. The article of manufacture of claim 18, wherein a first correctness rule specifies that invocation of a first function must follow an invocation of a second function in the program code, and the electronically readable medium is further configured with instructions for causing a processor to perform the steps comprising:

in counting an observance of the first correctness rule by a particular code instance, identifying program code that includes a sequence of instructions that includes an invocation of a particular first function following invocation of a particular second instruction; and

in counting a violation of the first correctness rule by the particular code instance, identifying program code that includes a sequence of instructions that includes an invocation of the particular second instruction without a previous invocation of the particular first function.

25. The article of manufacture of claim 18, wherein a first correctness rule specifies that data returned from a first function must be tested for a status indication, and the electronically readable medium is further configured with instructions for causing a processor to perform the steps comprising:

in counting an observance of the first correctness rule by a particular code instance, identifying program code that includes a sequence of instructions that includes an invocation of a particular first function and a subsequent test of data returned from the particular first function; and

in counting a violation of the first correctness rule by the particular code instance, identifying program code that includes a sequence of instructions that includes an invocation of a particular first function without a subsequent test of data returned from the particular first function.